Search: ○ The ACM Digital Library   ◉ The Guide

US Patent & Trademark Office

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Interprocedural Def-Use associations in C programs

Authors   Hemant D. Pande Siemens Corporate Research Inc., 755 College Rd. East, Princeton, NJ
                 William Landi   Department of Computer Science, Rutgers University, New Brunswick, New Jersey

Additional Information: references citings index terms collaborative colleagues

Tools and Actions: Discussions   Find similar Articles   Review this Article
                            Save this Article to a Binder   Display in BibTex Format

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1   Hiralal Agrawal , Joseph R. Horgan, Dynamic program slicing, ACM SIGPLAN Notices, v.25 n.6, p.246-256, Jun. 1990

2   D. Callahan, The program summary graph and flow-sensitive interprocedual data flow analysis, Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation, p.47-56, June 20-24, 1988, Atlanta, Georgia, United States

3   David R. Chase , Mark Wegman , F. Kenneth Zadeck, Analysis of pointers and structures, Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation, p.296-310, June 1990, White Plains, New York, United States

4   Anita L. Chow , Andres Rudmik, The design of a data flow analyzer, Proceedings of the 1982 SIGPLAN symposium on Compiler construction, p.106-113, June 23-25, 1982, Boston, Massachusetts, United States

5   B. G. Cooper. Ambitious data flow analysis of procedural programs. Master's thesis, University of Minnesota, May 1989.

6   Keith D. Cooper , Ken Kennedy, Efficient computation of flow insensitive interprocedural summary information, Proceedings of the 1984 SIGPLAN symposium on Compiler construction, p.247-258, June 17-22, 1984, Montreal, Canada

7   K. D. Cooper , K. Kennedy, Fast interprocedual alias analysis, Proceedings of the 16th ACM

PORTAL

Search: ● The ACM Digital Library ○ The Guide

Precise flow-insensitive May-Alias analysis

THE ACM DIGITAL LIBRARY   ℞ Feedback

Terms used Precise flow insensitive May Alias analysis

Sort results by   [relevance ▾]   ● Save results to a Binder   Try
                                             🔲 Search Tips   Try
                                             ☐ Open results in a new window

Display results [expanded form ▾]

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9
Best 200 shown

1 A schema for interprocedural modification side-effect analysis with pointer
Barbara G. Ryder, William A. Landi, Philip A. Stocks, Sean Zhang, Rita Altucher
March 2001   ACM Transactions on Programming Languages and Systems (TOPLA
Full text available: 📄 pdf(1.72 MB)   Additional Information: full citation, abstract, references, citing

> The first interprocedural modification side-effects analysis for C (MODC) that (
> on programs with general-purpose pointer usage is presented with empirical r
> algorithm schema corresponding to a family of MODC algorithms with two ind(
> pointer-induced aliases and a subsequent one for propagating interprocedural

2 An incremental flow- and context-sensitive pointer aliasing analysis
Jyh-shiarn Yur, Barbara G. Ryder, William A. Landi
May 1999   Proceedings of the 21st international conference on Software engine(
Full text available: 📄 pdf(1.29 MB)   Additional Information: full citation, references, citings, index t(

Keywords: incremental analysis, interprocedural pointer aliasing, interprocedu

3 Using static single assignment form to improve flow-insensitive pointer ana

Rebecca Hasti, Susan Horwitz

May 1998 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conferer and implementation, Volume 33 Issue 5

Full text available: pdf(958.17 KB)   Additional Information: full citation, abstract, references,

A pointer-analysis algorithm can be either flow-sensitive or flow-insensitive. V provides more precise information, it is also usually considerably more costly contribution of this paper is the presentation of another option in the form of provide a range of results that fall between the results of flow-insensitive and combines a flow-insensitive poi ...

4 Precise flow-insensitive may-alias analysis is NP-hard

Susan Horwitz

January 1997 ACM Transactions on Programming Languages and Systems (TOPL

Full text available: pdf(127.89 KB)   Additional Information: full citation, abstract, references, cit

Determining aliases is one of the foundamental static analysis problems, in pa problem is solved can affect the precision of other analyses such as live varial propagation. Previous work has investigated the complexity of flow-sensitive a that precise flow-insensitive may-alias analysis is NP-hard given arbitrary leve ...

Keywords: alias analysis, dataflow analysis, pointer analysis, static analysis

5 Precise and efficient integration of interprocedural alias information into dat

Michael Burke, Jong-Deok Choi

March 1992 ACM Letters on Programming Languages and Systems (LOPLAS), V

Full text available: pdf(499.90 KB)   Additional Information: full citation, abstract, references, citing

Data-flow analysis is a basis for program optimization and parallelizing transfc reference parameters at call sites generates interprocedural aliases which com been developed for efficiently computing interprocedural aliases. However, fac data-flow information has been mostly overlooked, although improper factorir data- ...

Keywords: alias analysis and optimization, data-flow analysis, interprocedural

6 ## Session 4: static program analysis: Searching for points-to analysis

Glenn Bruns, Satish Chandra

November 2002   ACM SIGSOFT Software Engineering Notes, Volume 27 Iss

Full text available: pdf(967.96 KB)   Additional Information: full citation, abstract, referenc

The complexity of points-to analysis is well understood, but the approximation efficiently are less well understood. In this paper we characterize points-to an program's state space. Reachability analysis can be performed approximately which certain basic program transformations have been applied. We show the in several existing points-to analysi ...

7 ## Data-flow analysis of program fragments

Atanas Rountev, Barbara G. Ryder, William Landi

October 1999 ACM SIGSOFT Software Engineering Notes , Proceedings of the 7th I conference held jointly with the 7th ACM SIGSOFT international sym engineering, Volume 24 Issue 6

Full text available: pdf(1.46 MB)   Additional Information: full citation, abstract, references, ci

Traditional interprocedural data-flow analysis is performed on whole programs analysis is not feasible for large or incomplete programs. We propose fragmen approach which computes data-flow information for a specific program fragme additional information available about the rest of the program. We describe tw flow-sensit ...

8 ## Precise interprocedural dataflow analysis via graph reachability

Thomas Reps, Susan Horwitz, Mooly Sagiv

January 1995 Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Prin

Full text available: pdf(1.51 MB)   Additional Information: full citation, abstract, references, ci

The paper shows how a large class of interprocedural dataflow-analysis proble polynomial time by transforming them into a special kind of graph-reachabilit the set of dataflow facts must be a finite set, and that the dataflow functions i operator (either union or intersection). This class of probable problems includ to—the classical separable problems (als ...

9 New results on the computability and complexity of points--to analysis

Venkatesan T. Chakaravarthy

January 2003 ACM SIGPLAN Notices , Proceedings of the 30th ACM SIGPLAN-SIGA programming languages, Volume 38 Issue 1

Full text available: pdf(423.68 KB) Additional Information: full citation, abstract, referen

Given a program and two variables p and q, the goal of points-to analysis is t execution of the program. This well-studied problem plays a crucial role in cor known to be undecidable when dynamic memory is allowed. But the result is l to be structures. We extend the result to show that, the problem remains und variables are allowed. O ...

Keywords: complexity, flow-insensitive, flow-sensitive, pointer analysis, unde

10 Session 4: static program analysis: Improving program slicing with dynamic

Markus Mock, Darren C. Atkinson, Craig Chambers, Susan J. Eggers

November 2002 ACM SIGSOFT Software Engineering Notes, Volume 27 Iss

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references

Program slicing is a potentially useful analysis for aiding program understandi programs are often too large to be generally useful. Imprecise pointer analyse this problem. In this paper, we use dynamic points-to data, which represents information, to obtain a bound on the best case slice size improvement that ci precision. Our experiments show that slice size ca ...

Keywords: dynamic analysis, points-to analysis, program slicing

11 Interprocedural pointer alias analysis

Michael Hind, Michael Burke, Paul Carini, Jong-Deok Choi

July 1999 ACM Transactions on Programming Languages and Systems (TOPLAS)

Full text available: pdf(502.42 KB) Additional Information: full citation, abstract, references, citi

We present practical approximation methods for computing and representing i written in a language that includes pointers, reference parameters, and recurs contributions: (1) a framework for interprocedural pointer alias analysis that h the program call graph while alias analysis is being performed; (2) a flow-sen analysis algorithm; (3 ...

Keywords: interprocedural analysis, pointer aliasing, program analysis

12 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam I

October 1999   ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN confe
systems, languages, and applications,   Volume 34 Issue 10

Full text available: pdf(1.85 MB)   Additional Information: full citation, abstract, references, ci

This paper presents a simple and efficient data flow algorithm for escape anal
determine (i) if an object can be allocated on the stack; (ii) if an object is acc
lifetime, so that synchronization operations on that object can be removed. W
for escape analysis, the connection graph, that is used to establish reachabilit
object ref ...

13 Efficient computation of flow insensitive interprocedural summary informati

Keith D. Cooper, Ken Kennedy

June 1984   ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium
Issue 6

Full text available: pdf(970.59 KB)   Additional Information: full citation, referen

14 Pointer analysis: haven't we solved this problem yet?

Michael Hind

June 2001   Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program
engineering

Full text available: pdf(199.83 KB)   Additional Information: full citation, abstract, references,

During the past twenty-one years, over seventy-five papers and nine Ph.D. th
analysis. Given the tomes of work on this topic one may wonder, “Have
With input from many researchers in the field, this paper describes issues rela
open problems.

15 Software analysis: a roadmap

Daniel Jackson, Martin Rinard

May 2000   Proceedings of the conference on The future of Software engineering

Full text available: pdf(1.51 MB)   Additional Information: full citation, references, citings, index terms

16 Double iterative framework for flow-sensitive interprocedural data flow anal

István Forgács

January 1994   ACM Transactions on Software Engineering and Methodology (TOS

Full text available: pdf(1.77 MB)   Additional Information: full citation, abstract, reference

Compiler optimization, parallel processing, data flow testing, and symbolic del data flow analysis. However, the live, reaching definition, and most summary intractable in the interprocedural case. A method is presented that reduces th of an algorithm that solves the problem in polynomial time. Either the resultir missing (or additional) resu ...

Keywords: data flow analysis, double iterative frameworks

17 Demand-driven pointer analysis

Nevin Heintze, Olivier Tardieu

May 2001 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conferei and implementation, Volume 36 Issue 5

Full text available: pdf(1.27 MB)   Additional Information: full citation, abstract, references, ci

Known algorithms for pointer analysis are “global&rdquo an exhaustive analysis of a program or program component. In demand-driven approach for pointer analysis. Specifically, we c flow-insensitive, subset-based, con text-insensitive points-to ar variables (a query), our analysis performs just enough compute sets for these query variables. ...

18 Technical papers: testing II: Fragment class analysis for testing of polymor

Atanas Rountev, Ana Milanova, Barbara G. Ryder

May 2003   Proceedings of the 25th international conference on Software engin

Full text available: pdf(1.13 MB) Publisher Site   Additional Information: full citation, a

Adequate testing of polymorphism in object-oriented software requires covera classes and target methods at call sites. Tools that measure this coverage nee coverage requirements. However, traditional whole-program class analysis cai programs. To solve this problem, we present a general approach for adapting operate on program fragments. Furthermore, ...

19 Static program analysis: Improving program slicing with dynamic points-to (

Markus Mock, Darren C. Atkinson, Craig Chambers, Susan J. Eggers

November 2002   Proceedings of the tenth ACM SIGSOFT symposium on Foundatic

Full text available: pdf(109.02 KB)   Additional Information: full citation, abstract, references,

Program slicing is a potentially useful analysis for aiding program understandi
programs are often too large to be generally useful. Imprecise pointer analyse
this problem. In this paper, we use dynamic points-to data, which represents
information, to obtain a bound on the best case slice size improvement that cɛ
precision. Our experiments show that slice size ca ...

Keywords: dynamic analysis, points-to analysis, program slicing

20 Cloning-based context-sensitive pointer alias analysis using binary decisior

John Whaley, Monica S. Lam

June 2004 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 confere
and implementation, Volume 39 Issue 6

Full text available: pdf(277.87 KB)   Additional Information: full citation, abstract, references,

This paper presents the first scalable context-sensitive, inclusion-based pointɛ
approach to context sensitivity is to create a clone of a method for every cont
context-insensitive algorithm over the expanded call graph to get *context-sen*
a clone for every acyclic path through a program's call graph, treating method
a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive
pointer analysis, program analysis, scalable

Results 1 - 20 of 200   Result page: **1**   2   3   4   5   6   7

The ACM Portal is published by the Association for Computing Machinery. C

Terms of Usage   Privacy Policy   Code of Ethics   Cont

Useful downloads: Adobe Acrobat   QuickTime   Windows Med

SIGPLAN-SIGACT symposium on Principles of programming languages, p.49-59, January 11-13, 1989, Austin, Texas, United States

8   Deborah S. Coutant, Retargetable high-level alias analysis, Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, p.110-118, January 01, 1986, St. Petersburg Beach, Florida

9   Phyllis G. Frankl , Elaine J. Weyuker, A data flow testing tool, Proceedings of the second conference on Software development tools, techniques, and alternatives, p.46-53, December 1985, San Francisco, California, United States

10   M. J. Harrold and M. L. Soffa. Interprocedural data flow testing. In Proceedings of the Third Testing, Analysis, and Verification Symposium, pages 158- 167, December 1989.

11   M. J. Harrold and M. L. Soffa. Computation of interprocedural definition and use dependencies. In Proceedings of the 1990 International Conference on Computer Languages, pages 297-306, 1990.

12   L. J. Hendren and A. Nicolau. Parallelizing programs with recursive data structures. In Proceedings of the 1989 International Conference on Parallel Processing, pages 49-56, August 1989.

13   S. Horwitz , P. Pfeiffer , T. Reps, Dependence analysis for pointer variables, Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation, p.28-40, June 19-23, 1989, Portland, Oregon, United States

14   S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. In Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation, pages 35-46, July 1988. SIGPLAN NOTICES, Vol. 23, No. 7.

15   N. Jones and S. Muchnick. Flow analysis and optimization of lisp-like structures, In S. Muchnick and N. Jones, editors, Program Flow Analysis: Theory and Applications, pages 102-131. Prentice Hall, 1979.

16   Bogdan Korel , Janusz Laski, Dynamic slicing of computer programs, Journal of Systems and Software, v.13 n.3, p.187-195, Nov. 1990

17   William Alexander Landi, Interprocedural aliasing in the presence of pointers, Rutgers University, New Brunswick, NJ, 1992

18   W. Landi and B. G. Ryder. Aliasing with and without pointers: A problem taxonomy. Center for Computer Aids for Industrial Productivity Technical Re- 148 port CAIP-TR-125, Rutgers University, September 1990.

19   William Landi , Barbara G. Ryder, Pointer-induced aliasing: a problem taxonomy, Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.93-103, January 21-23, 1991, Orlando, Florida, United States

20   W. Landi and B. G. Ryder. A safe approximate algorithm for interprocedural pointer aliasing. Laboratory for computer science research technical report, Rutgers University, August 1991. in preparation.

21   James Richard Larus , Paul Hilfinger, Restructuring symbolic programs for concurrent execution on multiprocessors, 1989

22   J. R. Larus and P. N. Hilfinger. Detecting conflicts between structure accesses. In Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation, pages 21-34, July 1988. SIGPLAN NOTICES, Vol. 23, No. 7.

23   Sying-Syang Liu and Abu-Bakr Taha. Interprocedural definition-use dependency analysis for

recursive procedures. Technical Report SERC-TR-42-F, Software Engineering Research Center, University of Florida, Gainesville, Florida, March 1990.

24   D. Lomet. Data flow analysis in the presence of procedure calls. Journal of Research and Development, 21(6):559-571, November 1977.

25   Eugene M. Myers, A precise inter-procedural data flow algorithm, Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.219-230, January 26-28, 1981, Williamsburg, Virginia

26   Thomas J. Ostrand. Data-flow testing with pointers and function calls. In Proceedings of the Pacific Northwest Software Quality Conference, October 1990.

27   H. D. Pande, W. Landi, and B. G. Ryder. Interprocedural reaching definitions in the presence of single level pointers. Siemens corporate research technical report, Siemens, 1991. in preparation.

28   Sandra Rapps , Elaine J. Weyuker, Selecting software test data using data flow information, IEEE Transactions on Software Engineering, v.11 n.4, p.367-375, April 1985

29   B. G. Ryder. Ismm: Incremental software maintenance manager. In Proceedings of the IEEE Computer Society Conference on Software Maintenance, pages 142-164, October 1989.

30   Martin Carroll , Barbara G Ryder, An incremental algorithm for software analysis, Proceedings of the second ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments, p.171-179, December 09-11, 1986, Palo Alto, California, United States

31   William E. Weihl, Interprocedural data flow analysis in the presence of pointers, procedure variables, and label variables, Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.83-94, January 28-30, 1980, Las Vegas, Nevada

32   Mark Weiser. Program slicing. IEEE Transactions on Software Engineering, SE-10(4) :352-357, July 1984.

33   Wuu Yang , Susan Horwitz , Thomas Reps, A program integration algorithm that accommodates semantics-preserving transformations, ACM SIGSOFT Software Engineering Notes, v.15 n.6, p.133-143, Dec. 1990

↑ CITINGS 9

Ruiqi Lian , Zhaoqing Zhang , Ruliang Qiao, Automatic generation of interprocedural data-flow analyzers and optimizers, Journal of Computer Science and Technology, v.17 n.6, p.708-717, November 2002

Thomas J. Ostrand , Elaine J. Weyuker, Data flow-based test adequacy analysis for languages with pointers, Proceedings of the symposium on Testing, analysis, and verification, p.74-86, October 08-10, 1991, Victoria, British Columbia, Canada

Gregg Rothermel , Mary Jean Harrold, A framework for evaluating regression test selection techniques, Proceedings of the 16th international conference on Software engineering, p.201-210, May 16-21, 1994, Sorrento, Italy

Monica Hutchins , Herb Foster , Tarak Goradia , Thomas Ostrand, Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria, Proceedings of the 16th international conference on Software engineering, p.191-200, May 16-21, 1994, Sorrento, Italy

William Landi , Barbara G. Ryder, A safe approximate algorithm for interprocedural aliasing, ACM SIGPLAN Notices, v.27 n.7, p.235-248, July 1992

William Landi , Barbara G. Ryder , Sean Zhang, Interprocedural modification side effect analysis with pointer aliasing, ACM SIGPLAN Notices, v.28 n.6, p.56-67, June 1993

H. D. Pande , W. A. Landi , B. G. Ryder, Interprocedural Def-Use Associations for C Systems with Single Level Pointers, IEEE Transactions on Software Engineering, v.20 n.5, p.385-403, May 1994

William Landi , Barbara G. Ryder, A safe approximate algorithm for interprocedural pointer aliasing, ACM SIGPLAN Notices, v.39 n.4, April 2004

Hong Zhu , Patrick A. V. Hall , John H. R. May, Software unit test coverage and adequacy, ACM Computing Surveys (CSUR), v.29 n.4, p.366-427, Dec. 1997

⋔ **INDEX TERMS**

Primary Classification:
 D. Software
  ⤷ **D.3** PROGRAMMING LANGUAGES
   ⤷ **D.3.2** Language Classifications

    ⤷ **Nouns:** C

Additional Classification:
 D. Software
  ⤷ **D.2** SOFTWARE ENGINEERING
   ⤷ **D.2.5** Testing and Debugging
    ⤷ **Subjects:** Testing tools (e.g., data generators, coverage testing)
  ⤷ **D.3** PROGRAMMING LANGUAGES
   ⤷ **D.3.3** Language Constructs and Features
    ⤷ **Subjects:** Control structures
   ⤷ **D.3.4** Processors
    ⤷ **Subjects:** Optimization

 F. Theory of Computation
  ⤷ **F.2** ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY
   ⤷ **F.2.2** Nonnumerical Algorithms and Problems
    ⤷ **Subjects:** Computations on discrete structures
  ⤷ **F.3** LOGICS AND MEANINGS OF PROGRAMS
   ⤷ **F.3.3** Studies of Program Constructs
    ⤷ **Subjects:** Control primitives

⋔ **Collaborative Colleagues:**
 William Landi: Rita Z. Altucher
          Ramkrishna Chatterjee
          Hemant D. Pande
          Atanas Rountev
          Barbara G. Ryder
          Sean Zhang
 Hemant D. Pande: William Landi
          Barbara G. Ryder

PORTAL

Search: ● The ACM Digital Library   ○ The Guide

Alias analysis <and> generating alias tree <and> equivalence class <and>

THE ACM DIGITAL LIBRARY   Feedback

Terms used Alias analysis and generating alias tree and equivalence class and definition use

Sort results by	relevance ▼	● Save results to a Binder	Try
		▣ Search Tips	Try
Display results	expanded form ▼	☐ Open results in a new window	

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9
Best 200 shown

1 A schema for interprocedural modification side-effect analysis with pointer
Barbara G. Ryder, William A. Landi, Philip A. Stocks, Sean Zhang, Rita Altucher
March 2001   ACM Transactions on Programming Languages and Systems (TOPLA

Full text available: ▣ pdf(1.72 MB)   Additional Information: full citation, abstract, references, citing:

The first interprocedural modification side-effects analysis for C (MODC) that (
on programs with general-purpose pointer usage is presented with empirical r
algorithm schema corresponding to a family of MODC algorithms with two ind
pointer-induced aliases and a subsequent one for propagating interprocedural

2 Program decomposition for pointer aliasing: a step toward practical analyse
Sean Zhang, Barbara G. Ryder, William Landi
October 1996 ACM SIGSOFT Software Engineering Notes , Proceedings of the 4th
Foundations of software engineering, Volume 21 Issue 6

Full text available: ▣ pdf(1.12 MB)   Additional Information: full citation, abstract, references, ci

Pointer aliasing analysis is crucial to compile-time analyses for languages witl
as C), but many aliasing methods have proven quite costly. We present a tecl
a program to allow separate, and therefore possibly different, pointer aliasing
independent parts of the program. This decomposition enables exploration of
and precision. We also present a new, effi ...

3 Equivalence analysis: a general technique to improve the efficiency of data pointers

Donglin Liang, Mary Jean Harrold

September 1999 ACM SIGSOFT Software Engineering Notes , Proceedings of the 1ξ Program analysis for software tools and engineering, Volume 24

Full text available: pdf(874.77 KB)   Additional Information: full citation, abstract, references,

Existing methods to handle pointer variables during data-flow analyses can m time and space because the data-flow analyses must store and propagate larg by dereferences of pointer variable. This paper presents *equivalence analysis,* efficiency of data-flow analyses in the presence of pointers. The technique ide memory locations accessed by a ...

Keywords: alias analysis, data-flow analysis

4 Alias annotations for program understanding

Jonathan Aldrich, Valentin Kostadinov, Craig Chambers

November 2002 ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN con programming, systems, languages, and applications, Volume 37]

Full text available: pdf(306.14 KB)   Additional Information: full citation, abstract, references,

One of the primary challenges in building and evolving large object-oriented s between objects. Unexpected aliasing can lead to broken invariants, mistaken surprising side effects, all of which may lead to software defects and complica presents AliasJava, a capability-based alias annotation system for Java that m code, enabling developers to reason more effec ...

Keywords: aliasing, aliasjava, encapsulation, java, ownership types, type infei

5 Interprocedural pointer alias analysis

Michael Hind, Michael Burke, Paul Carini, Jong-Deok Choi

July 1999   ACM Transactions on Programming Languages and Systems (TOPLAS)

Full text available: pdf(502.42 KB)   Additional Information: full citation, abstract, references, cit

We present practical approximation methods for computing and representing i written in a language that includes pointers, reference parameters, and recurs contributions: (1) a framework for interprocedural pointer alias analysis that h the program call graph while alias analysis is being performed; (2) a flow-sen analysis algorithm; (3 ...

Keywords: interprocedural analysis, pointer aliasing, program analysis

6 Interprocedural may-alias analysis for pointers: beyond k-limiting

Alain Deutsch

June 1994 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 confere and implementation, Volume 29 Issue 6

Full text available: pdf(1.36 MB)   Additional Information: full citation, abstract, references, citing

Existing methods for alias analysis of recursive pointer data structures are bas k-limiting, and store-based (or equivalently location or region-based) approxin elements of recursive data structures. Although notable progress in inter-proc accomplished, very little progress in the precision of analysis of recursive poir

7 Escape analysis for Java^{TM}: Theory and practice

Bruno Blanchet

November 2003   ACM Transactions on Programming Languages and Systems (TOF

Full text available: pdf(884.21 KB)   Additional Information: full citation, abstract, reference:

Escape analysis is a static analysis that determines whether the lifetime of da

paper first presents the design and correctness proof of an escape analysis for interprocedural, context sensitive, and as flow-sensitive as the static single as object fields are analyzed in a flow-insensitive manner. Since Java is an impei assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronizatioi

8 Equivalence analysis and its application in improving the efficiency of progr

Donglin Liang, Mary Jean Harrold

July 2002   ACM Transactions on Software Engineering and Methodology (TOSEM)

Full text available: pdf(457.78 KB)   Additional Information: full citation, abstract, referenc

Existing methods for handling pointer variables during dataflow analyses can i time and space because the data-flow analyses must store and propagate larg by dereferences of pointer variables. This article presents *equivalence analysi* efficiency of data-flow analyses in the presence of pointer variables. The techi among the memory locations ...

Keywords: Alias analysis, data-flow analysis, program slicing

9 An interval-based approach to exhaustive and incremental interprocedural

Michael Burke

July 1990   ACM Transactions on Programming Languages and Systems (TOPLAS)

Full text available: pdf(4.43 MB)   Additional Information: full citation, abstract, references, citing

We reformulate interval analysis so that it can be applied to any monotone da
problems of flow-insensitive interprocedural analysis. We then develop an incr
can be applied to the same class of problems. When applied to flow-insensitiv
the resulting algorithms are simple, practical, and efficient. With a single upd
accommodate any sequence of pr ...

10 A compiler framework for speculative analysis and optimizations

Jin Lin, Tong Chen, Wei-Chung Hsu, Pen-Chung Yew, Roy Dz-Ching Ju, Tin-Fook

May 2003 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conferer
and implementation, Volume 38 Issue 5

Full text available: pdf(323.88 KB)   Additional Information: full citation, abstract, references,

Speculative execution, such as control speculation and data speculation, is an
performance. Using edge/path profile information or simple heuristic rules, ex
adequately incorporate and exploit control speculation. However, very little ha
compiler frameworks to incorporate and exploit data speculation effectively in
beyond instruction scheduling. This paper proposes a ...

Keywords: data speculation, partial redundancy elimination, register promotio
weak update

11 The path-wise approach to data flow testing with pointer variables

Delia I. S. Marx, Phyllis G. Frankl

May 1996 ACM SIGSOFT Software Engineering Notes , Proceedings of the 1996 A(
on Software testing and analysis, Volume 21 Issue 3

Full text available: pdf(941.63 KB)   Additional Information: full citation, abstract, referen

This paper describes a new approach to performing data flow testing on progr;
tool based on this approach. Our technique is based on the observation that, ι
we can determine which dereferenced pointers are aliased whenever control r;
particular path. Furthermore, we can group together paths which behave simil
expressions. The resulting test requirements ...

12 Using types to analyze and optimize object-oriented programs

Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss

January 2001   ACM Transactions on Programming Languages and Systems (TOPL

Full text available: pdf(414.51 KB)   Additional Information: full citation, abstract, references,

Object-oriented programming languages provide many software engineering k performance cost. Object-oriented programs make extensive use of method in both of which are potentially costly on modern machines. We show how to use techniques that reduce the costs of these features in Modula-3, a statically typ compiler performs type-based alias analysis to ...

Keywords: alias analysis, classes and objects, method invocation, object orien elimination

13 Storeless semantics and alias logic

Marius Bozga, Radu Iosif, Yassine Laknech

June 2003   ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN worksh semantics-based program manipulation, Volume 38 Issue 10

Full text available: pdf(270.73 KB)   Additional Information: full citation, abstract, referen

Pioneering work has been done by Jonkers [18] to define a semantics of point abstract in the sense of ignoring low-level aspects such as dangling pointers a principles of such storeless semantics from a logical point of view, first definir characterize heap structures up to isomorphism. Second, we extend this langu allows to express regular properties of unboun ...

Keywords: heap models, total correctness, weakest precondition

14 Role analysis

Viktor Kuncak, Patrick Lam, Martin Rinard

January 2002   ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGA programming languages, Volume 37 Issue 1

Full text available: pdf(2.27 MB)   Additional Information: full citation, abstract, referer

We present a new *role system* in which the type (or *role*) of each object deper other objects, with the role changing as these relationships change. Roles cap properties and provide useful information about how the actions of the progra role system enables the programmer to specify the legal aliasing relationship: may play, the ...

15 Abstract description of pointer data structures: an approach for improving tl imperative programs

Joseph Hummel, Laurie J. Hendren, Alexandru Nicolau

September 1992   ACM Letters on Programming Languages and Systems (LOPLAS

Full text available: pdf(1.23 MB)   Additional Information: full citation, abstract, references, citing

Even though impressive progress has been made in the area of optimizing an the application of similar techniques to programs using pointer data structure: which have a small number of well-defined properties, pointers can be used tc which exhibit a much larger set of properties. The diversity of these structure: data structures cannot be effect ...

16 Type-based alias analysis

Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss

May 1998 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conferel and implementation, Volume 33 Issue 5

Full text available: pdf(1.66 MB)   Additional Information: full citation, abstract, references, ci

This paper evaluates three alias analyses based on programming language ty compatibility to determine aliases. The second extends the first by using addii field names. The third extends the second with a flow-insensitive analysis. Alt types to disambiguate memory references, none evaluates its effectiveness. V evaluations of type-based alias analyses for Mod ...

17 Automatic generation and management of interprocedural program analyse

Kwangkeun Yi, Williams Ludwell Harrison

March 1993   Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Princi

Full text available: pdf(1.32 MB)   Additional Information: full citation, abstract, references, cil

We have designed and implemented an interprocedural program analyzer gen automate the generation and management of semantics-based interprocedura target languages. System Z is based on the abstract interpretation framework specification of an abstract interpreter. The output is a C code for the specifiec system ...

18 Ownership types for safe programming: preventing data races and deadloc
Chandrasekhar Boyapati, Robert Lee, Martin Rinard
November 2002 ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN con
programming, systems, languages, and applications, Volume 37]
Full text available: pdf(459.57 KB)   Additional Information: full citation, abstract, references,

This paper presents a new static type system for multithreaded programs; we
guaranteed to be free of data races and deadlocks. Our type system allows pr
fixed number of equivalence classes and specify a partial order among the eqt
statically verifies that whenever a thread holds more than one lock, the threa(
order.Our system also allows programmer ...

Keywords: data races, deadlocks, encapsulation, ownership types

19 Pointer analysis: haven't we solved this problem yet?
Michael Hind
June 2001 Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program
engineering
Full text available: pdf(199.83 KB)   Additional Information: full citation, abstract, references,

During the past twenty-one years, over seventy-five papers and nine Ph.D. th
analysis. Given the tomes of work on this topic one may wonder, “Have
With input from many researchers in the field, this paper describes issues rela
open problems.

20 Ownership, encapsulation and the disjointness of type and effect
Dave Clarke, Sophia Drossopoulou
November 2002 ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN con
programming, systems, languages, and applications, Volume 37]
Full text available: pdf(475.72 KB)   Additional Information: full citation, abstract, references,

Ownership types provide a statically enforceable notion of object-level encaps
computational effects to support reasoning about object-oriented programs. Tl
control and effects reporting. Based on this type system, we codify two formal
and the disjointness of computational effects. The first can be used to prove tl
never lead to aliases, while the ...

Keywords: aliasing, encapsulation, ownership types, type-and-effects systems

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7